AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A computer-implemented system configured to place a controllable amount of stress on a server that is running an application[~~that~~] in order to test load ~~loads a~~ the server by dynamically randomly generating, on a per iteration basis, each request of a test load based on a predefined set of weighted user characteristics such that the percentages of user characteristics of the totality of the requests statistically corresponds to the weighted percentages in the user characteristics, in order to simulate a diverse population of users accessing the application without using upfront determination of user characteristics for simulated users, the system comprising:

a processor; and

memory storing the following:

a profile characteristic data store comprising the predefined set of weighted user characteristics;

[~~a~~]one or more load simulators, interfaced to the profile characteristic data store, each having a dynamic load adjuster component that, for each iteration of the test load, dynamically randomly generates~~adjusts~~ user characteristics for a request based on percentage weightings in the predefined set of weighted user characteristics, wherein the percentage weightings statistically designate ~~at least in part on a browser type, for~~ distribution of user characteristics ~~thereof~~ as a percentage of total requests sent to ~~a~~ the server ~~being load tested~~ such that whereas each request is individually generated randomly, as the number of iterations increases, the load simulator generates a totality of requests that statistically corresponds to the weightings in the profile characteristic data store,

a load coordinator component that dynamically evaluates the current distribution of the test load relative to a desired test load and adjusts the intensity and distribution of the requests, including increasing the requests per second to a predetermined level; and

a performance monitor component that monitors performance of the server as the rate of requests is increased, so the load capacity of the server can be determined.

2 – 3. (Canceled)

4. (Currently Amended) The system of claim [2]1, the predefined set of user characteristics ~~comprises~~ comprising at least one of: network connections, browser types, and load patterns.

5. (Currently Amended) The system of claim [2]1, wherein the predefined set of user characterstics are statistically determined based on web log records.

6. (Currently Amended) The system of claim [2]1, wherein the predefined set of user characteristics are predetermined in a single user profile.

7. (Canceled)

8. (Previously Presented) The system of claim 1, further comprising an artificial intelligence component.

9. (Previously Presented) The system of claim 1, further comprising a closed loop control to enable a continual and sustained rate of requests to the server.

10 – 15. (Canceled)

16. (Currently Amended) A computer-implemented method for load testing a server, whereby a controllable amount of stress may be placed on the server via an application running on the server, the method comprising:

assigning weights to user characteristics in a user profile;

dynamically randomly generating, for each iteration of a plurality of iterations in a test load, a request according to percentage weightings in the weighted user characteristics, wherein the percentage weightings are statistical parameters that designated distribution of user characteistics as a percentage of total requests, such that whereas each request is generated randomly, as the number of iterations increases, the load simulator generates a totality of requests with user characteristics that statistically corresponds to the weightings in the profile characteristic data store;

~~adjusting the user characteristics based on one or more browser types during the testing of the server; and~~

~~distributing the user characteristics as a percentage of total requests sent to the server~~

dynamically evaluating, upon ending the iteration of the test load, the current test load relative to a desired test load and adjusting the intensity and distribution of the requests, including one of either creating a new request if the desired load is greater than the current load, or reducing the current test load by one if the current load rises above the desired load.

17 – 19. (Canceled)

20. (Previously Presented) The method of claim 16, further comprising controlling a rate of loading via a feedback loop control.

21. (Canceled)

22. (New) The system of claim 1, wherein the one or more load simulators comprise a plurality of load simulators.

23. (New) The computer-implemented method of claim 16, wherein the user characteristics are stored in no more than a single user profile.

24. (New) A machine-implemented system that dynamically stresses a server by providing an adjustable rate of requests per second (RPS) to conduct stress testing, failure predictions, and capacity planning, the system comprising:

an execution engine that generates a scenario that loads the server via a plurality of requests, the plurality of requests dynamically adjusted based on a user profile having weighted characteristics that comprises at least a browser type therein, wherein user characteristics are distributed as a percentage of total requests, and wherein the execution engine comprises

a data store containing the user profile, including the weighted user characteristics;

a scheduler;

a queueing mechanism that retrieves data from the data store based on a received signal input from the scheduler and places the request data in a queue and sorts requests within the queue according to a predetermined time function for execution, wherein the retrieved request data is randomly selected based on the weighted user characteristics;

a sending component that reads and sends a sorted request from the queue upon receiving an input from the scheduler based upon a rate determined by the scheduler in order to provide a desired RPS;

a feedback loop which provides closed loop control to enable the system to provide a continual and sustained rate of requests, wherein the feedback loop provides an input to the scheduler that is calculated based on the difference between an actual RPS and a target RPS, wherein the scheduler, based on the input, adjusts the rate of requests according to the target RPS.